A-007

辞書の共有化を用いた Range Coder 法の性能評価
Performance Evaluation of Range Coding Using Shared Dictionaries

安井 秀太[1)]　喜田 拓也[1)]
Shuta Yasui　Takuya kida

1　はじめに

データ圧縮は、主にデータの保存コストや通信コストを低減するために用いられ、これまでに様々な手法が提案されている [1]。なかでも、ハフマン符号に代表されるエントロピー符号化は、データをモデル化した後の後段の符号化で用いられることの多い、重要なデータ圧縮技術である。

算術符号 [2] を改良した Range Coder 法 [6] は、使用メモリ量と圧縮速度の観点において優れたエントロピー符号化として知られている。単体で用いた場合の圧縮率は、ハフマン符号と同程度以上、算術符号以下である。

Range Coder 法は、記号の出現頻度を辞書として圧縮データに含める必要がある。よって、サイズの小さいファイル群に対してデータ圧縮を行うと、ファイル毎に個別の辞書を保存することになり、圧縮率の観点から効率が悪い。同種のファイルに対して共通の辞書を使って圧縮することができれば、全体的な圧縮率の向上が期待できる。その一方で、ファイルの性質と一致しない辞書を用いた場合には圧縮率が悪化する可能性もある。

このように辞書を共有化する考え方は古くからあり、特に辞書式圧縮法において共有辞書を用いて圧縮率を向上させる手法 [3, 4] が提案されている。また、Google が開発した Brotli [5] なども共有辞書のアイデアに基づいている。これは、ブラウザとサーバで辞書を共有することでウェブコンテンツを効率よく圧縮し、ページ通信コストを大幅に低減する技術である。

本稿では、Range Coder 法において辞書を共有化した際の圧縮性能について予備的な実験を行い、その結果について考察する。

2　準備

本節では、まず算術符号について概説する。その後に、Range Coder 法の符号化アルゴリズムについて説明を行う。

2.1　算術符号

算術符号の符号化は、各記号の出現確率の割合で区間 $[0, 1)$ を分割することから始まる。たとえば、記号 a, b, c が確率 $0.6, 0.3, 0.1$ であったとき、区間 $[0, 1)$ を 3 つの区間 $[0, 0.6)$, $[0.6, 0.9)$, $[0.9, 1)$ に分割する。分けられた区間は元の記号 a, b, c にそれぞれ対応づけられる。たとえば、入力テキストの最初の文字が b の場合は、区間 $[0.6, 0.9)$ に対応づけられる。そして、この区間をさらに各記号の出現確率の割合で分割する。すなわち、$[0.6, 0.78)$, $[0.78, 0.87)$, $[0.87, 0.9)$ となり、これらはそれぞれ ba, bb, bc に対応する区間となる。よって、入力テキストの先頭が bc の場合、対応する区間は $[0.87, 0.9)$ である。この手続きを繰り返し、データ全体を区間 $[0, 1)$ 内のある小区間 $[p, q)$ に対応づける。最後は、区間を表す数 p, q に共通する接頭辞となる有理数を二進符号化することでデータ全体の符号化を行う。算術符号の復号には、辞書として元のデータ長と各記号の出現確率の情報が必要となる。また、実際に算術符号を実行する際は、区間の実数値表現のために、桁あふれや丸め誤差などを考慮した工夫が必要となる。浮動小数点数を扱うため計算処理コストが高く、また特許の問題もあり、実用される場面は少ない。

2.2　Range Coder 法

Range Coder 法は算術符号化の問題を克服し、現実的に利用しやすくした圧縮方法である。アルファベット Σ 上の長さが n の入力系列 $a_1 a_2 \cdots a_n$（$a_i \in \Sigma$）に対する符号化アルゴリズムは以下のとおりである。

1. 各記号 $a \in \Sigma$ の出現確率を $p(a)$ とする。
2. 十分に大きな整数 K を取り、区間 $[0, K]$ を考える。
3. この区間を各記号の出現確率の割合で区分けする。
4. $i = 1, 2, \ldots, n$ について、以下を繰り返す：
   (a) 現在の区間のうち、記号 a_i に対応する部分の区間 $[L, R]$ を選択する。
   (b) あらかじめ定めた定数 m に対し、選択された区間の幅 $R - L$ が K/m 以下になった場合、次の区間を $[mL, mR]$ に拡大する。
   (c) mL の値が 4 バイト整数の桁からあふれる場合、桁あふれした上位部分を符号語列の一部として出力する。
   (d) 拡大後の区間を、再び出現確率に応じて区分けする。
5. 最後に選択された区間 $[L, R]$ の L の値を符号語列の末尾に出力して終了する。

Range Coder 法では、区間の計算をすべて整数で行い、さらに出力もバイト単位で行うため、算術符号と比べて非常に高速である。

3　実験

サイズの小さなファイル群に対して、共有辞書を用いて圧縮した場合と、ファイルを個別に圧縮した場合との圧縮率を比較する実験を行った。

今回、国会議事録検索システム[a)]から、特定の単語をキーワードとしてヒットした会議録のテキストをファイルとして保存したものを用いた。ファイルの日本語エンコードはすべて UTF-8 である。キーワードとして、「東日本大震災」「消費税」「ガソリン」「子育て」「防災」「年金」「雇用」「医療」「エネルギー」の 10 個を選択し、それぞれヒットした会議録を 30 ファイルずつ無作為に取得した。取得したファイルの最小サイズは 5,915 byte で、最大サイズは 445,356 byte、平均 177,641 byte で

1）北海学園大学大学院工学研究科電子情報生命工学専攻

a) https://kokkai.ndl.go.jp/#/

カテゴリ	元サイズ	個別に圧縮	共有辞書利用
東日本大震災	1,634	1,014 (62.10%)	1,012 (**61.92**%)
消費税	1,877	1,162 (61.89%)	1,159 (**61.72**%)
ガソリン	1,904	1,184 (62.16%)	1,181 (**62.01**%)
子育て	2,199	1,357 (61.70%)	1,355 (**61.61**%)
防災	1,530	944 (61.71%)	943 (**61.63**%)
年金	2,014	1,238 (61.45%)	1,236 (**61.36**%)
雇用	2,038	1,252 (61.42%)	1,248 (**61.26**%)
医療	1,767	1,091 (61.73%)	1,088 (**61.56**%)
エネルギー	1,724	1,065 (61.74%)	1,063 (**61.64**%)
憲法	1,653	1,026 (62.08%)	1,025 (**62.01**%)

表 1　カテゴリごとに 10 ファイル分を合計した圧縮率の比較（単位は Kbyte。個別に圧縮は辞書を含む）

あった。このデータを、キーワード別にカテゴリー分けされたテキスト群として用いた。各カテゴリーにおいて、20 個のファイルを共有辞書作成用のデータとし、残り 10 個のファイルを圧縮するテスト用データとする。

指定した辞書を用いて Range Coder 法で圧縮を行うプログラムは、広井誠氏のホームページ[b] を参考に Python で実装したものを用意した。このプログラムは、バイトコードを記号として取り扱うため、任意の形式のファイルに適用することができる（すなわち記号の種類数は 256 である）。また、各記号の出現頻度を 2byte で表現するため、辞書のサイズは $256 \times 2 = 512$byte となる。出現しない記号にも区間を割り当てる必要があるため、すべての記号の最小頻度を 1 にしている。また、2byte が表現できる非負整数の最大値 $2^{16} - 1 = 65535$ を超える頻度の記号がある場合は、その最大頻度が 65535 になるように記号全体の頻度を正規化している。

表 1 に、カテゴリごとの圧縮率比較の結果を示す。表中の各数値は、カテゴリー毎に 10 ファイル分のサイズの合計（単位は Kbyte）である。個別にファイルを圧縮したほうは、辞書分のデータ（各 512 バイト）も圧縮ファイルに含んでいる。一方、共有辞書を利用して圧縮したほうは、圧縮ファイルに辞書は含まれていない。丸かっこの中は元データサイズに対する圧縮データサイズの比率（圧縮率）を表しており、数値が小さいほど良く圧縮されていることを示している。

表 1 からは、共有辞書を用いて圧縮してもほぼ圧縮率にデメリットがなく、個別の圧縮ファイルに辞書が含まれていない分、わずかにファイルサイズが小さくなっていることが読み取れる。このことから、共有辞書と各ファイルの個別の辞書とにあまり差がないことが推測できる。

表 2 に、「東日本大震災」のテスト用ファイル 10 ファイル分について、各ファイル個別の辞書と各カテゴリーの共有辞書との KL ダイバージェンスを示す。ここでは、辞書から記号の確率分布を求めており、ファイル側の確率分布を P、共有辞書側の確率分布を Q としたときの $D(P||Q)$ を求めている。また、各行の最小値を太字で示している。

表 2 から読みとれるように、これら 10 個のファイルについては、カテゴリー分けとはほぼ無関係にどの共有辞書とも KL ダイバージェンスの値が小さい。表は省く

b) Algorithms with Python: データ圧縮 (中編), https://www.nct9.ne.jp/m_hiroi/light/pyalgo36.html

	東日本大震災	消費税	ガソリン	子育て	防災	年金	雇用	医療	エネルギー	憲法
file1	5.7	6.8	7.0	**4.0**	4.8	4.6	5.8	6.1	6.4	5.4
file2	3.1	3.9	3.0	2.6	2.6	2.7	**2.4**	2.9	2.5	3.6
file3	8.3	7.0	7.6	**6.1**	8.2	6.9	7.1	6.8	7.4	8.4
file4	17.6	13.6	17.1	**11.9**	17.6	15.5	16.1	15.4	17.3	15.8
file5	5.4	5.2	5.9	4.8	6.1	4.6	4.6	**4.4**	5.5	4.5
file6	16.5	12.4	16.2	**10.4**	15.9	14.4	14.9	14.6	15.8	14.8
file7	11.2	11.1	11.5	9.5	**9.4**	9.9	11.6	11.8	11.9	10.6
file8	3.3	6.0	5.9	**2.8**	3.6	3.2	3.2	3.2	3.5	3.3
file9	41.9	43.2	47.5	40.9	41.1	43.5	44.6	44.0	44.9	**37.0**
file10	12.2	12.7	13.6	13.6	13.6	12.8	13.4	12.7	14.0	**10.6**

表 2　「東日本大震災」のテスト用ファイル個別の辞書 P と各カテゴリーの共有辞書 Q との KL ダイバージェンス $D(P||Q)$（数値の単位は 10^{-3}）

が、他のカテゴリーのテスト用ファイルについても同様の傾向が見られた。すなわち、今回のカテゴリー分けでは、データにあまり明確な偏りが見られず、全体で「日本語 UTF-8 のデータ」という一つのカテゴリーとしても差支えないことがわかる。なお、表 2 の結果を元に、この 10 個のファイルそれぞれを最も KL ダイバージェンスの小さかったカテゴリーの共有辞書を用いて圧縮したところ、ごくわずかながら圧縮率の改善（61.92% から 61.90%）が見られた。

4　さいごに

本稿では、サイズの小さなファイル群に対して Range Coder 法でデータ圧縮する際、共有辞書を用いることの効果について調査を行った。今回用いたデータでは、共有辞書を用いて圧縮してもほぼ圧縮率が悪くならず、辞書を個別に保存する必要がない分、わずかにファイルサイズが小さくなることが確認できた。

今後の課題としては、ファイル群に対して辞書の距離が近いもので自動的にカテゴリー分けを行う方法について検討することが挙げられる。

参考文献

[1] D. Salomon and G. Motta, Handbook of Data Compression, Springer Publishing Company, Incorporated, 2009.

[2] Ian H. Witten, Radford M. Neal, and John G. Cleary, Arithmetic coding for data compression, Communications of the ACM, Volume 30, Issue 6, pp.520-540, June 1987.

[3] R. Wan and A. Moffat, Block Merging for Off-Line Compression, Proc. of the 13th Annual Symposium on Combinatorial Pattern Matching (CPM '02), Springer-Verlag, Berlin, Heidelberg, pp.32-41, July 2002.

[4] K. Sekine, H. Sasakawa, S. Yoshida, and T. Kida, Variable-to-Fixed-Length Encoding for Large Texts Using Re-Pair Algorithm with Shared Dictionaries. 2013 Data Compression Conference (DCC2013), p.518, March, 2013.

[5] J. Alakuijala and Z. Szabadka, Brotli Compressed Data Format, Internet Engineering Task Force (IETF), Request for Comments: 7932, Category: Informational, Google, Inc. ISSN: 2070-1721, July 2016.

[6] G. Martin, Range encoding: an algorithm for removing redundancy from a digitised message, Procedure of Video & Data Recording Conference, Southampton, UK, July 24-27, 1979.